Object Oriented Programming Lab

Final Project - “Contact Book Application”

Muhammad Yousaf

|  |
| --- |
| **Topics:**   * **Making an Object Oriented** Application * Use of **Inheritance, Polymorphism, Operator Overloading & templates** * **Three file structure** * **File Handling** * Everything you studied till now |

# Project Brief

This project will provide you an opportunity to test and review all the concepts that you studied during the semester. The goal for the final project is to extend the Mini Project: **ContactsBook** **application**, with some new features to make it more realistic and useful!

Extending the mini project might be rather easy using Windows Forms instead of console. Let's discuss!

The project requirement is to be developed on console. Graphical User Interface with Windows Forms is a bonus and recommended for top groups, you can discuss with me! Currently, your project will be only usable through one computer, which is yours. You can continue working on it to **turn it into a website or an Android app this summer ☺ , interesting, no?**

# Policies

1. Project will be done in groups of 2
2. You really need to understand this document. And strictly follow the instructions provided to you
3. Plagiarism policies you already know. ☺
4. Your Viva will be coding based in the Lab and you’ll be asked to modify/extend any function of this project individually.
5. Your final marks will be dependent upon the percentage you get in viva and quality of code. Even if your marks in code are 100%, viva and code quality marks will affect these marks directly.

**HAPPY CODING**

# Code Design Guidelines

First of all see the following Google C++ coding standards guidelines:

**For naming**: <https://google.github.io/styleguide/cppguide.html#Naming>

**Formatting**: <https://google.github.io/styleguide/cppguide.html#Formatting>

**Function Arguments**: <https://google.github.io/styleguide/cppguide.html#Inputs_and_Outputs>

**Classes**: <https://google.github.io/styleguide/cppguide.html#Classes>

|  |  |
| --- | --- |
| *Duplication* | As instructed and taught in Mini-Project, you must minimize code duplication: **Clean coding, minimum code duplication is a requirement and is a must! If you need help removing duplication, you can take help from me.** |
| *Function Names* | |  |  |  | | --- | --- | --- | | Use PascalCase: Examples | | | | MyFunction | |  | | SortArray |  | | GetRollNumbe | | | |
| *Class Names* | Use PascalCase |
| *Variable Names* | Use snake\_case For example: roll\_number first\_name |
| File Names | Use Snake case:  Filenames should be all lowercase and include underscores (\_) Examples:   |  |  | | --- | --- | | my\_useful\_class |  | | myusefulclass\_test | | |
| *Header Files Usage* | You will use Header files (3-file-structure to structure your project). Properly separate concepts keeping in mind Single Responsibility Principle as I mentioned in class. |

|  |  |
| --- | --- |
| *Class Per File* | Maximum one class should reside in one header file, CPP files shall have implementation of maximum one class. |
| *Use const variables where necessary* | Use const where necessary examples:   1. Array sizes 2. Don’t hardcode strings like Menuitems, such as “1. Add Item”. Instead create a const std::string variable in your class and initialize it in line like:   const std::string ADD\_ITEM = “1. Add Item”;   1. Make getters constant and any other read only functions like equal, compare etc. |
| **Also, you must create the destructors to free all the memory allocated using new.** | |
| **Follow the Rule of Three** [**Rule of three (C++ programming)**](https://en.wikipedia.org/wiki/Rule_of_three_(C%2B%2B_programming)#:~:text=The%20rule%20of%20three%20)[**- Wikipedia**](https://en.wikipedia.org/wiki/Rule_of_three_(C%2B%2B_programming)#:~:text=The%20rule%20of%20three%20) | |
| **Single Responsibility Principle: Your functions shall do only one thing they are made for. Like functions should not be doing cin or cout stuff but returning values. Just try to keep the cin/cout to the main function/some main class or functions specially made specifically for printing or input such as an overloaded cin operator.**  **Example: Search function can just return a list of matched contacts along with the size of the list (a struct can be made to return multiple things from a function).** | |

**Do exception handling where needed and gracefully print errors.**

Task-I: Getting the starter code

Download the starter code file from classroom.

# Task-II: Understanding the code

In this project you will complete an incomplete application to store contacts. You are required to fully understand the code written by me and the structure of the code. There are 7 files provided to you. You will have to complete the missing parts in this code.

|  |  |
| --- | --- |
| Address.h | This file contains the data member declarations of Address class.  Address class defines an Address that has a street, country, house data members etc. |
| Address.cpp | All the functions declared in Address.h are implemented in this file using Scope resolution operator **::** |
| Contact.h | This file contains the data member |
|  | declarations of Contact class.  This class models a contact number that may have a name, phone and email and an **Address object**. |
| Contact.cpp | This file contains implementation for member functions of Contact class |
| ContactsBook.h | ContactsBook defines the list of Contacts as an array of **Contact objects.**  ContactsBook class also defines member function to perform operations like adding a contact to list, printing contacts etc (you will complete their implementations) |
| ContactsBook.cpp | Implementations for ContactsBook class member functions go here |
| Main.cpp | You will write driver code in this class |

# Task-III: Problems

*Note: Strictly follow the meaningful names, naming convention (function name like* ***function\_name****, variable name like* ***variable\_name****, class name like* ***ClassName****) and maintain proper spacing and indentation.*

**Problem I**

Complete the tasks mentioned in the comments in the code provided to you

## Problem II: Driver code, Main function

Write a menu-driven program that offers the following options

1. Create a contacts list form given size (input size from user)
2. Add New Contact
3. Merge Duplicates
4. Store To File (input file name)
5. Load From File (input file name)
6. Print Contacts Sorted (input choice, fist\_name or last\_name)
7. Print Contacts
8. Search contacts (all three choices)
9. Display Count of Contacts

Run the following scenarios using the input/output given below.

**Scenario 1**

1. Create a ContactsBook object with size 5
2. Create 5 Contacts, set first\_name, last\_name, Address and all other attributes

**Note: You will need to set Address by creating an Address object for each contact. (You should create a reusable function that takes input(cin) Home, Street etc and returns an Address object).**

**Bonus: Even better if you operator overloading for input of contacts and its address.**

1. Print the contacts on screen, sorted by First Name
2. Now, print unsorted contacts (create a new simple printing function in ContactsBook class)
3. Take a screenshot that you will submit

### Scenario 2: Continue with Scenario 1

1. Display the count of contacts
2. Add two more contacts (also Address for each contact)
3. Print the total number of contacts stored in the list Output shall be 7.
4. Take a screenshot

**Scenario 3: Add 5 contacts with 1 duplicate, display all the contacts sorted. Now, run duplicate merge and display results**

**Scenario 4: Show the working of store contacts and load contacts**

# Submission on Google Classroom

**Note: You should submit the complete code as project both zip file (header+cpp ) and MS-word file.**